UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/735,678	12/16/2003	Mark Justin Moore	56162.000382	3580

21967 7590 08/20/2008

HUNTON & WILLIAMS LLP
INTELLECTUAL PROPERTY DEPARTMENT
1900 K STREET, N.W.
SUITE 1200
WASHINGTON, DC 20006-1109

EXAMINER	
ARCOS, CAROLINE H	
ART UNIT	**PAPER NUMBER**
2195	

MAIL DATE	DELIVERY MODE
08/20/2008	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	10/735,678	MOORE ET AL.	
	Examiner	Art Unit	
	CAROLINE ARCOS	2195	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE *3* MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *16 December 2003*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-22* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-22* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☒ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *16 December 2003* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date *05/17/2004 and 10/04/2004*.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. Claims 1-22 are pending for examination.

Specification

2. The abstract of the disclosure is objected to because it contains more than 150 words.

3. The specification is objected to as failing to provide proper antecedent basis for the claimed subject matter of "computer readable medium" in claims 12-22. See 37 CFR 1.75(d)(1) and MPEP § 608.01(o).

Correction is required.

Claim Rejections - 35 USC § 101

4. 35 U.S.C. 101 reads as follows:

> Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.

5. Claims 12-22 are rejected under 35 U.S.C. 101 because the claimed invention is directed to non-statutory subject matter.

6. As per claim 12-18, 19-21 and 22, the claimed computer readable medium is software per se as tit is not statutory embodied on any sort of physical (hardware) medium. the claims recite instructions performing function which is software functions/module without claiming associated computer hardware required for storing and execution. Software alone is directed to a non statutory subject matter.

Claim Rejections - 35 USC § 112

7. The following is a quotation of the second paragraph of 35 U.S.C. 112:

> The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.

8. Claims 1-22 are rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention.

a. The following terms lacks antecedent basis:

i. The type of thread- claim 7.

ii. The previously executing thread, the array of threads –claim 8.

b. The claim language of the following claims is not clearly understood:

i. As per claim1, lines 5-6, it is not clearly understood whether each time slot is allocated to a particular thread as claimed in lines 3-4 or each time slot is allocated to a queue of threads. Line 10, it is unclear whether the "array of threads" is the same as "queue of threads" and if they are not the same, what is the relation between them? Line 11, it is not clear what is the criteria for suspending a currently executing thread? (i.e. time slice expired or higher priority thread arrived) also it is not clearly understood whether the currently executing thread is from the queue of threads or the array of thread or are they the same. Line 14, it is not clear what is the criteria of appending the suspended currently executing thread?(i.e. to get a second chance to run) line 16, it is not clearly understood

what is meant by non-empty time slot? Line 19, it is unclear what is the criteria of appending any content of the indexed time slot to the array of threads.

ii. As per claim 12, it has the same deficiency as claim 1.

iii. As per claim 6, line 2, it is unclear if the currently executing thread is suspended for the second time or is this a different currently executing thread? Lines 6-7, it is unclear how the determination of the emptiness of the array of threads is done?

iv. As per claim 17, it has the same deficiency as claim 6.

v. As per claim 7, line 6, it is not clearly understood how it is determined that the interrupt request immediate CPU allocation? Line 11, it is unclear what is meant by "type" of currently executing thread (priority level?) Line 15, it is not clearly understood how many times the currently executing thread is being suspended.

vi. As per claim 18, it has the same deficiency as claim 7.

vii. As per claim 8, it has the same deficiency as claim 1 since it has similar limitation. Furthermore line 13, it is unclear whether the priority is assigned to all the threads including the one in the queue of threads and the array of queue. Line 15; it is unclear what is the criteria of removing for each of the plurality of circular arrays, each queue of threads for the indexed time slot? Line 33, it is unclear what is the previously executing thread? (i.e. it is the suspended thread?)

viii. As per claim 19, it has the same deficiency as claim 19.

ix. As per claim 11, it has similar limitation as well as deficiency as claim 1.

Claim Rejections - 35 USC § 103

9. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

10. Claims 1-22 are rejected under 35 U.S.C. 103(a) as being unpatentable over Moore (US 2004/0205753), in view of Huynh et al. (US 5,386,561).

11. As per claim 1, Moore teaches the invention substantially as claimed including a method for scheduling thread execution, comprising:

maintaining a circular array structure having a plurality of time slots therein, wherein each of the plurality of time slots corresponds to a timeslice during which CPU resources are allocated to a particular thread (abs., lines 1-16; wherein the events/ functions are threads to be performed);

configuring each time slot in the circular array to include a queue of threads scheduled for execution during that time slot (abs. Lines 9-17);

maintaining a pointer index for referencing one time slot in the circular array and whereby advancement through the circular array is provided by advancing the pointer index (fig. 2, 200; par.[0028]) ;

calculating a next time slot during which the currently executing thread should next

resume execution (fig. 2, 200; wherein time(t + p) is the next sequential non empty time slot after time(t) slot where the threads that are executing in time t will be resuming);

identifying a next sequential non-empty time slot (fig. 2, 200);

updating the pointer index to point to the identified next sequential non-empty time slot (fig.2, 200).

12. Moore doesn't explicitly teach maintaining an array of threads requesting immediate CPU resource allocation;

suspending a currently executing thread;

appending the suspended currently executing thread to the queue of threads scheduled for execution at the calculated time slot;

appending any contents of the indexed time slot to the array of threads requesting immediate CPU resource allocation;

removing the thread at the top of the array of threads requesting immediate CPU resource allocation; and

activating the thread at the top of the array of threads requesting immediate CPU resource allocation.

13. However, Huynh teaches maintaining an array of threads requesting immediate CPU resource allocation (col. 5, lines 63-68; col. 6, lines 1-6);

suspending a currently executing thread (col. 6, lines 17-29);

appending the suspended currently executing thread to the queue of threads scheduled for

execution at the calculated time slot(col. 6, lines 20-21).

appending any contents of the indexed time slot to the array of threads requesting immediate CPU resource allocation (col. 6, lines 60-66)

removing the thread at the top of the array of threads requesting immediate CPU resource allocation (col. 6, lines 60-66; where removing the thread at the top of the array is scheduling the highest priority thread); and

activating the thread at the top of the array of threads requesting immediate CPU resource allocation(col. 6, lines 60-66; where activating the thread at the top of the array is scheduling/executing the highest priority thread).

14. It would have been obvious to one of ordinary skill in the art at the time the invention was made to combine Moore and Huynh that Huynh teaching would improve scheduling techniques and system performance by giving priority to threads with higher importance over lower priority threads that it is not that important for the system to be scheduled.

15. As per claim 2, Moore teaches that each timeslice is between 10 and 100 microseconds (par. [0016], lines 13-15).

16. As per claim 3, the combined teaching doesn't explicitly teach that the array of threads requesting immediate CPU resource allocation includes a first-in-first-out (FIFO) structure. However, it would have been obvious to one of ordinary skill in the art at the time the invention was made that one would be knowledgeable that FIFO or LIFO structure is a matter of design

choice of the developer.

17. As per claim 4, Huynh teaches the step of suspending a currently executing thread includes: suspending the currently executing thread upon expiration of a current timeslice (col. 6, lines 16-20).

18. As per claim 5, Huynh teaches the step of suspending a currently executing thread includes: receiving a self-suspend request from the currently executing thread (col. 6. lines 21-25).

19. As per claim 6, Moore teaches that advancing the index pointer by one time slot (fig. 2, 200; par. [0028]).

20. Moore doesn't explicitly teach that suspending the currently executing thread;

removing a list of any threads to be executed at the indexed time slot and appending them to the array of threads requesting immediate CPU resource allocation;

determining whether the array of threads requesting immediate CPU resource allocation is empty;

returning to the step of advancing the index pointer by one slot if it is determined that the array of threads requesting immediate CPU resource allocation is empty; and

removing and activating the thread at the top of the array of threads requesting immediate CPU resource allocation if it is determined that the array of threads requesting

immediate CPU resource allocation is not empty.

21. However, Huynh teaches suspending the currently executing thread(col. 6, lines 17-29; col. 6, lines 60-66);

removing a list of any threads to be executed at the indexed time slot and appending them to the array of threads requesting immediate CPU resource allocation(col. 6, lines 60-66; wherein executing the highest priority thread first the lower priority thread that were executing is appending the lower priority threads that suppose to be executing to be executing after the highest priority threads.);

determining whether the array of threads requesting immediate CPU resource allocation is empty (col. 6, lines 60-68);

returning to the step of advancing the index pointer by one slot if it is determined that the array of threads requesting immediate CPU resource allocation is empty (col. 7, lines 1-7); and

removing and activating the thread at the top of the array of threads requesting immediate CPU resource allocation if it is determined that the array of threads requesting immediate CPU resource allocation is not empty(col. 6, lines 60-66; where activating the thread at the top of the array is scheduling/executing the highest priority thread).

22. It would have been obvious to one of ordinary skill in the art at the time the invention was made to combine Moore and Huynh that Huynh teaching would improve scheduling techniques and system performance by giving priority to threads with higher importance over

lower priority threads that it is not that important for the system to be scheduled.

23. As per claim 7, Moore teaches calculating a next time slot during which the currently executing thread should next resume execution (fig. 2, 200; wherein time(t + p) is the next sequential non empty time slot after time(t) slot where the threads that are executing in time t will be resuming);

24. Moore doesn't explicitly teach receiving an external event interrupt requesting CPU resource allocation for a new thread;

determining whether the external event interrupt is requesting immediate CPU resource allocation;

appending the new thread to a queue on a time slot on the circular array if it is determined that the external event interrupt is not requesting immediate CPU resource allocation;

determining the type of thread currently executing; activating the new thread if no thread is currently executing;

performing the following steps if it is determined that a non-idle thread is currently executing:

suspending the currently executing thread;

appending the currently executing thread to the end of the array of threads requesting immediate CPU resource allocation; and

activating the new thread;

performing the following steps if it is determined that an idle thread is currently

executing:

suspending the currently executing thread; and

activating the new thread.

25. However, Huynh teaches appending the new thread to a queue on a time slot on the circular array if it is determined that the external event interrupt is not requesting immediate CPU resource allocation(col.6, lines 1- 29; col.6. lines 60-66; wherein if the threads that need immediate attention is less priority than the executing thread, the thread will be place to the queue corresponding to its priority)

determining the type of thread currently executing (col.6, lines 1-29);

performing the following steps if it is determined that a non-idle thread is currently executing:

suspending the currently executing thread (col.6, lines 16-29);

appending the currently executing thread to the end of the array of threads requesting immediate CPU resource allocation (col.6, lines 20-21); and

activating the new thread (col.6, lines 25-30);

performing the following steps if it is determined that an idle thread is currently executing(col.6, lines 21-25 ; wherein an idle thread is a thread that doesn't do any work that cannot run no more waiting for resource):

suspending the currently executing thread(col.6, lines 16-29); and

activating the new thread(col.6, lines 25-30).

26. The combined teaching doesn't explicitly teach activating the new thread if no thread is currently executing. It would have been obvious to one of ordinary skill in the art at the time the invention was made to conclude that new thread if no thread is currently executing since there is no other threads that has greater priority running which improve system performance and scheduling techniques.

27. As per claim 8, Moore teaches the invention substantially as claimed including a method for scheduling thread execution, comprising:

maintaining a plurality of circular array structures associated with a plurality of threads, each having a plurality of time slots therein, wherein each of the plurality of time slots corresponds to a timeslice during which CPU resources are allocated to a particular thread (abs., lines 1-16; par. [0016]; par. [0024], lines 12-16; wherein the events/ functions are threads to be performed);

configuring each time slot in each of the circular arrays to include a queue of threads scheduled for execution during that time slot (abs. Lines 9-17);

maintaining at least one pointer index for referencing one time slot in each of the circular arrays whereby advancement through the circular arrays is provided by advancing the pointer index (par. [0016]; fig.2, 200; par. [0028]);

incrementing the index pointer by one slot (par. [0016]; fig.2, 200; par. [0028]);

28. Moore doesn't explicitly teach plurality of circular array structures associated with a plurality of discrete thread priorities,

maintaining an array of threads requesting immediate CPU resource allocation for each of the plurality of circular arrays;

assigning each thread to be executed a specific priority;

removing, for each of the plurality of circular arrays, each queue of threads for the indexed time slot;

appending each removed thread to the array of threads requesting immediate CPU resource allocation associated with its respective circular array;

determining whether the array of threads requesting immediate CPU resource allocation associated with a first circular array is non-empty; proceeding to a next circular array if the array of threads requesting immediate CPU resource allocation is empty;

extracting its top thread if the array of threads requesting immediate CPU resource allocation is non-empty;

determining whether a priority of the top thread is greater than a priority of the currently executing thread; calculating a time for next execution of the top thread if it is determined that the priority of the top thread is not greater than the priority of the currently executing thread;

performing the following steps if it is determined that the priority of the top thread is greater than a priority of the currently executing thread: suspending the currently executing thread; activating the top thread; and

calculating the time of next execution for the previously executing thread; determining whether each of the array of threads requesting immediate CPU resource allocation associated with each of the circular arrays has been processed; and

proceeding to the next array of threads requesting immediate CPU resource allocation if

it is determined that not all arrays of threads requesting immediate CPU resource allocation have been processed.

29. However, Huynh teaches plurality of circular array structures associated with a plurality of discrete thread priorities (col.6, lines 1-15),

maintaining an array of threads requesting immediate CPU resource allocation for each of the plurality of circular arrays (col. 5, lines 63-68; col. 6, lines 1-6);

assigning each thread to be executed a specific priority (col. 6, lines 2-6);

removing, for each of the plurality of circular arrays, each queue of threads for the indexed time slot (col. 6, lines 60-66)

appending each removed thread to the array of threads requesting immediate CPU resource allocation associated with its respective circular array (col. 6, lines 60-66)

determining whether the array of threads requesting immediate CPU resource allocation associated with a first circular array is non-empty(col.6, lines 60-66);

proceeding to a next circular array if the array of threads requesting immediate CPU resource allocation is empty (fig.2; col.6, lines 25-29; col.6, lines 67 wherein, if there is no higher priority process(high priority queue is empty), there will be no preemption of the currently executing process

extracting its top thread if the array of threads requesting immediate CPU resource allocation is non-empty (fig.2; col. 6, lines 65-66);

determining whether a priority of the top thread is greater than a priority of the currently executing thread(6, lines 65-66);

performing the following steps if it is determined that the priority of the top thread is greater than a priority of the currently executing thread:

suspending the currently executing thread (col. 6, lines 17-21);

activating the top thread (fig.2, col.6, lines 60-66; wherein scheduling the execution of the highest priority thread is activating the thread that has the highest priority which is the first one on the highest priority queue); and

calculating the time of next execution for the previously executing thread (col. 6, lines 16-20);

determining whether each of the array of threads requesting immediate CPU resource allocation associated with each of the circular arrays has been processed(col.6, lines 16-21; and

proceeding to the next array of threads requesting immediate CPU resource allocation if it is determined that not all arrays of threads requesting immediate CPU resource allocation have been processed (col.6, lines 1-30; wherein scheduling does depending the priority level and since the threads with the same priority are in the same queue, the next thread to be processed would be from the same priority level that need immediate attention and if all threads within the same priority have been processed, the next priority level queue is serviced).

30. The combined teaching doesn't explicitly teach calculating a time for next execution of the top thread if it is determined that the priority of the top thread is not greater than the priority of the currently executing thread. However, it would have been obvious to one of ordinary skill in the art at the time the invention was made to conclude from Huynh teaching that calculating a time for next execution of the top thread if it is determined that the priority of the top thread is

not greater than the priority of the currently executing thread by summing the time quantum to service all the higher priority threads/queues until it come the turn of the priority level of the threads that need immediate resource allocation which improve scheduling techniques and system performance that will allow one to fine tune the system based on the result of the calculation.

31. As per claim 9, the combined teaching doesn't explicitly teach that each of the plurality of circular arrays corresponds to one of four assigned priority levels: a non-real-time priority; a soft-real-time priority; a hard-real-time priority; and a critical-real-time priority.

32. However, it would have been obvious to one of ordinary skill in the art at the time the invention was made to conclude from Moore and Huynh teaching that that each of the plurality of circular arrays corresponds to one of four assigned priority levels: a non-real-time priority; a soft-real-time priority; a hard-real-time priority; and a critical-real-time priority (Moore: par.[0024], lines 12-16) which allow the system to perform all type of different priority depending on how important the execution of the task to the system.

33. As per claim 10, Moore teaches calculating a next time slot during which the currently executing thread should next resume execution (fig. 2, 200; wherein time(t + p) is the next sequential non empty time slot after time(t) slot where the threads that are executing in time t will be resuming);

34. Moore doesn't explicitly teach receiving an external event interrupt requesting CPU resource allocation for a new thread;

determining whether the external event interrupt is requesting immediate CPU resource allocation;

appending the new thread to a queue on a time slot on the circular array if it is determined that the external event interrupt is not requesting immediate CPU resource allocation;

determining whether a priority of the new thread is greater than a priority of the currently executing thread;

appending the new thread to the end of the array of threads requesting immediate CPU resources for the associated priority if it is determined that the priority of the new thread is not greater than the priority of the currently executing thread; and

performing the following steps if it is determined that the priority of the new thread is greater than the priority of the currently executing thread:

suspending the currently executing thread;

calculating the time for next execution for the currently executing thread

appending the currently executing thread to array associated with the calculated time slot; and

activating the new thread.

35. However, Huynh teaches appending the new thread to a queue on a time slot on the circular array if it is determined that the external event interrupt is not requesting immediate CPU resource allocation (col.6, lines 1- 29; col.6. lines 60-66; wherein if the threads that need immediate attention is less priority than the executing thread, the thread will be place to the

queue corresponding to its priority)

determining whether a priority of the new thread is greater than a priority of the currently executing thread (col. 6, lines 60-66);

appending the new thread to the end of the array of threads requesting immediate CPU resources for the associated priority if it is determined that the priority of the new thread is not greater than the priority of the currently executing thread(col.6, lines 1- 29; col.6. lines 60-66; wherein if the threads that need immediate attention is less priority than the executing thread, the thread will be place to the queue corresponding to its priority); and

performing the following steps if it is determined that the priority of the new thread is greater than the priority of the currently executing thread:

suspending the currently executing thread (col. 6, lines 17-20);

calculating the time for next execution for the currently executing thread(col. 6, lines 16-20);

appending the currently executing thread to array associated with the calculated time slot (col.6, lines 20-21); t; and

activating the new thread (fig.2, col.6, lines 60-66; wherein scheduling the execution of the highest priority thread is activating the thread that has the highest priority which is the first one on the highest priority queue);

36. The combined teaching doesn't explicitly teach receiving an external event interrupt requesting CPU resource allocation for a new thread and determining whether the external event interrupt is requesting immediate CPU resource allocation. However, it would have been obvious

to one of ordinary skill in the art at the time the invention was made to conclude from Moore and Huynh teaching that that interrupts are threads with priority the one with higher priority than the currently executing thread can preempt (interrupt) the currently executing thread and preemption the currently executing thread is a interrupt with higher priority than the executing thread which improve scheduling techniques and system performance by giving priority to threads with higher importance over lower priority threads that it is not that important for the system to be scheduled.

37. As per claim 11, Moore teaches the invention substantially as claimed including a method for scheduling thread execution, comprising:

maintaining a circular array structure having a plurality of time slots therein, wherein each of the plurality of time slots corresponds to a timeslice during which CPU resources are allocated to a particular thread(abs., lines 1-16; wherein the events/ functions are threads to be performed);

configuring each time slot in the circular array to include a queue of threads scheduled for execution during that time slot (abs. Lines 9-17);

maintaining a pointer index for referencing one time slot in the circular array and whereby advancement through the circular array is provided by advancing the pointer index(fig. 2, 200; par.[0028]) ;

calculating a next time slot during which a currently executing thread should next resume execution(fig. 2, 200; wherein time(t + p) is the next sequential non empty time slot after time(t) slot where the threads that are executing in time t will be resuming);

identifying a next sequential non-empty time slot(fig. 2, 200);

updating the pointer index to point to the identified next sequential non-empty time slot(fig. 2, 200);

38. Moore doesn't explicitly teach that maintaining an array of threads requesting immediate CPU resource allocation;

appending the currently executing thread to the queue of threads scheduled for execution at the calculated time slot;

appending any contents of the indexed time slot to the array of threads requesting immediate CPU resource allocation;

removing the thread at the top of the array of threads requesting immediate CPU resource allocation;

maintaining execution of the currently executing thread for the following time slot if it is determined that the thread at the top of the array of threads requesting immediate CPU resource allocation is identical to the currently executing thread;

suspending a currently executing thread; and activating the thread at the top of the array of threads requesting immediate CPU resource allocation if it is determined that the thread at the top of the array of threads requesting immediate CPU resource allocation is not identical to the currently executing thread;

39. However, Huynh teaches maintaining an array of threads requesting immediate CPU resource allocation(col. 5, lines 63-68; col. 6, lines 1-6);

appending the currently executing thread to the queue of threads scheduled for execution

at the calculated time slot(col. 6, lines 20-21);

appending any contents of the indexed time slot to the array of threads requesting immediate CPU resource allocation (col. 6, lines 60-66; wherein executing the highest priority thread first the lower priority thread that were executing is appending the lower priority threads that suppose to be executing to be executing after the highest priority threads.);

determining whether the thread at the top of the array of threads requesting immediate CPU resource allocation is identical to the currently executing thread(col.6, lines 60-62);

removing the thread at the top of the array of threads requesting immediate CPU resource allocation (col. 6, lines 60-66; where removing the thread at the top of the array is scheduling the highest priority thread);

maintaining execution of the currently executing thread for the following time slot if it is determined that the thread at the top of the array of threads requesting immediate CPU resource allocation is identical to the currently executing thread(col. 6, lines 16-29 wherein being identical is having the same level of priority);

suspending a currently executing thread(col.6, lines 17-21); and

activating the thread at the top of the array of threads requesting immediate CPU resource allocation if it is determined that the thread at the top of the array of threads requesting immediate CPU resource allocation is not identical to the currently executing thread(col. 6, lines 17-29; col. 6, lines 60-66 wherein the test of not being identical of the level of priority).

40. It would have been obvious to one of ordinary skill in the art at the time the invention was made to combine Moore and Huynh that Huynh teaching would improve scheduling

techniques and system performance by giving priority to threads with higher importance over lower priority threads that it is not that important for the system to be scheduled.

41. As per claims 12-22, they are the computer-readable medium claim of the method claims 1-11. Therefore they are rejected under the same rational.

Conclusion

42. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

(US 6964046 B1) teaches System and method for scheduling a future event.

43. Any inquiry concerning this communication or earlier communications from the examiner should be directed to CAROLINE ARCOS whose telephone number is (571)270-3151. The examiner can normally be reached on Monday-Thursday 7:00 AM to 5:30 PM.

44. If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Meng-Ai An can be reached on 571-272-3756. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

45. Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished

applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/Caroline Arcos/
Examiner, Art Unit 2195

/Li B. Zhen/
Primary Examiner, Art Unit 2194